TrMenu

主页

介绍

TrMenu 是一款发布于 Oct 4th, 2019 的 Bukkit 高级菜单系统

相比市面上同类菜单插件，拥有丰富功能的同时，还具有强大高效的特点，广受国内外大量用户好评

当前已基于 Kotlin 大量改善和完全重写，使用超强 Bukkit-API 拓展 TabooLib5

功能

- **版本兼容，**支持 1.8-1.16, 低版本可用 ID/DataValue
- **数量不限，**菜单的数量没有上限，你可以制作无限制的独立高级菜单
- **高级布局，**直观可视化的制作菜单，现已支持多字符布局
- **多页菜单，**你可以利用布局轻松制作多页菜单
- **虚拟菜单，**完全基于数据包的虚拟容器 & 物品，更安全
- **玩家容器，**玩家背包 4*9 的槽位制作菜单，支持多页
- **动态标题，**所有菜单均支持周期性更新的动态标题
- **菜单事件，**为菜单的开启、关闭事件执行动作反应
- **周期任务，**自定义无限制个菜单周期任务
- **内置脚本，**配置多个自定义脚本并快速调用
- **高级绑定，**支持绑定到正则匹配的命令、多个物品特征或是快捷动作
- **显示材质，**插件支持各种头颅, CustomModelData, NBT 及 HDB 等
- **动态效果，**图标支持使用动态数量、动态发光效果，动态 NBT 等
- **动态图标，**轻松配置动态的材质、名称、Lore和槽位，支持独立更新周期
- **图标交互，**基于数据包，完整支持包括数字键的总计超过 22 种点击类型
- **执行动作，**超过 40+ 种动作支持，一步到位实现效果
- **动作选项，**每个动作均支持配置独立的选项，如延时、条件、概率和遍历
- **三元反应，**可以使用多个三元反应，由条件、通过动作组和拒绝动作组构成
- **条件图标，**每个图标可配置带条件的优先级子图标
- **图标继承，**子图标可以选择继承默认图标的显示属性以提高效率
- **脚本编译，**条件基于脚本编译返回结果，大部分脚本自动预编译缓存

- **智能条件，**原创的条件表达式写法，轻松入门上手，如 hasMoney.100
- **输入捕获，**支持多个参数的高级捕获器
- **命令传参，**开启命令的参数传入菜单作为变量，轻松打造插件级 GUI
- **元素变量，**为玩家增删改 Meta 值，且可以当变量使用
- **模板功能，**将物品放入容器，快速创建菜单配置
- **自动重载，**菜单编辑自动重载，即时可见
- **RGB 颜色，**完整支持 1.16+ 颜色代码, 如 &{FFFFFF}, &{256,256,256}
- 忽略大小写及多种写法的节点
- 原创易懂的节点逻辑，详细文档 & 轻松入门
- 代码开源，长期维护，提供开发者 API
- And much more ...

版本

- TrMenu v1.x
  - 普通免费版本
  - 停止更新、支持
- TrMenu v2.x
  - 现役最新版本
  - 付费开源

> 本教程完全基于 TrMenu v2.x
>
> 旧版本 Wiki：http://trmenu.trixey.cn/

开始

新版

性能改进

TrMenu v2 基于 Kotlin 完全重写，注重每一块的优化，将性能发挥至极致

完全基于数据包

纯发包容器 & 物品

高效灵活

去掉旧版部分繁琐的配置，化繁为简，为高效而生

新增诸多可拓展功能，灵活性大增

缓存脚本

大多数脚本将被智能缓存，提高效率

超级简单

没有比这更简单的菜单

更好的布局

布局不再为单字符所限, 现在可以通过 `` 来支持字符串图标布局

内置脚本

支持在菜单内配置可快速调用的脚本

周期任务

每个菜单可配置多个独立的周期任务，开启菜单后执行，关闭后取消

三元反应

所有执行动作的地方均可使用多个三元动作组，更方便灵活

不用优先级图标也能轻松做高级功能！

智能条件表达式

v1

```
player.hasPermission("demo") && %vault_eco_balance% >= 500
```

v2

```
hasPerm.demo and hasMoney.500
```

更多点击类型

完整支持数字键 1~9 和 切换副手键

并且支持快速配置多个点击类型触发的动作

更强捕获器

支持高级捕获器写法，多个参数

更强物品绑定

支持绑定物品特征，不仅局限于 Lore

更快的头颅

读取在线玩家 NMS 皮肤材质 & 异步联网获取离线玩家材质，不卡服

物品仓库

存储并在材质中调用一些物品

更多挂钩

支持 Cronus，SkinsRestorer & 持续增加中...

防频繁点击

每个菜单支持设置独立的防频繁点击延时

独立的更新周期

图标现在可以为材质、名称、Lore、槽位配置独立的更新周期

轻松实现更优质的动画效果

玩家容器

玩家背包 4*9 容器可作为拓展槽位，布局菜单图标

同时也支持多页

更强参数

参数在翻页等操作中给予保留，支持配置默认参数补全

参数不再为空格所限，支持用 `` 括起的字符串

打开命令

打开命令支持直接指定页码

元素变量

支持设置删改玩家独立的 Meta 值, 可作为变量快速调用

存储变量

支持设置删改玩家独立的 Data 值，可作为变量快速调用，支持存储

漏洞修复

修复了大量 TrMenu v1 遗留的问题，包括图标属性错乱，优先级刷新出错等

RGB 颜色

完整支持 1.16+ 的 Hex、RGB 颜色，&{FFFFFF} 或 &{256,256,256} 均支持

开发者 API

更完善的 API 以及 MenuFactory 快速构建基于数据包的 GUI

购买

> 付费购买本插件前，请先确认你的服务端核心、版本是否兼容

价格

- 89.9 RMB

折扣

- 50% OFF
  - 公益服或正版服 服主
  - **必须现有 MCBBS 过审宣传帖**
  - **正在开发的公益服务器不享受折扣**

渠道

- （推荐）爱发电付费
- （不推荐）国外资源平台
  - SpigotMC
  - Songoda
- 直接联系开发者转账

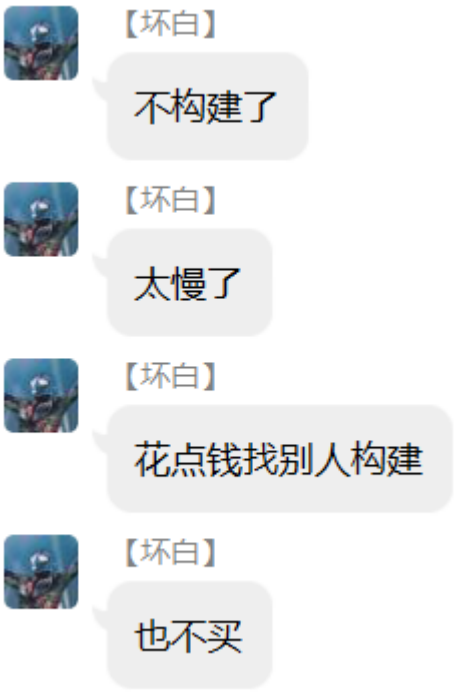
条款

- 你不得以任何形式向任何人分发本插件的任何部分

- 你不得声称本插件的任何部分为你的作品
- 你不得以任何理由申请退款
- 你不得未经授权转载本插件的任何信息

我为什么要购买？

- TrMenu v2.x 是付费插件的同时，在 Github 上完全开源
- 我相信本插件的品质配得上这笔大多数服主都能承担起的钱，
- 您如果实在**不愿意**支持这一份工作，也可以选择自行编译使用
- 但我将没有义务为你提供任何帮助亦或采纳建议



- **购买** 仅仅是对作者和项目的支持，可以获得售后和支持、采纳建议
- 而对于同上图类似的人，
- 请一定记得我们提供免费下载最新版的自动构建 ，免得多**浪费**您一分钱

安装

需求

> TrMenu 需要在联网条件下载安装

服务端	版本
Spigot	1.8-1.16
Paper	1.8-1.16
Akarin	×
Arclight (Forge)	√
CatServer (Forge)	√
Mohist (Forge)	×

安装

- 将 **TrMenu.jar** 丢进 plugins 目录
- 重新启动服务器

配置

文件

lang/zh_CN.yml

TLocale 语言文件, 你可以编辑大多数插件的消息

menus

默认的菜单加载目录

items.yml

物品仓库存储功能

settings.yml

插件的设置文件

设置

- 使用文本编辑器打开 settings.yml，你可以在此编辑插件的相关设置
- 插件将自动监听保存改动，重新载入

settings.yml

```yaml
#
# 插件的相关选项
#
Options:
  # 是否启用调试模式 (将打印更多信息到控制台)
  Debug: false
  # 使用的语言文件
  # 可用： zh_CN, zh_TW, en_US
  Locale: zh_CN
  # 是否隐藏插件启动时的 Logo
  Hide-Logo: false
  # 联网加载头颅材质是否使用 Mojang API
  # 不启用则默认使用 https://api.minetools.eu/
  Skull-Mojang-API: false

#
# 自定义加载菜单的文件或目录
# 支持多级子目录itemRepo
#
Load-Menu-Files:
  - 'plugins/CustomMenusFolder'

#
# 动作相关设置
#
Actions:
  # 指定取消捕获器的内容，支持正则
  Catcher-Cancel-Words:
    - '(?i)cancel|quit|exit|end|stop'
    - '取消|退出'

#
# 菜单物品相关
#
Item:
  # 默认物品名称的颜色
  Default-Color-Name: '&7'
  # 默认物品描述的颜色
  Default-Color-Lore: '&7'

#
# 快捷操作执行动作
#
Shortcuts:
  Offhand: []
  Sneaking-Offhand:
    - condition: 'hasPerm.trmenu.shortcut'
      execute: 'open: Example'
```

```yaml
  Right-Click-Player: []
  Sneaking-Right-Click-Player: []
  PlayerInventory-Border-Left: []
  PlayerInventory-Border-Right: []

#
# 注册命令执行动作
# (增改命令需要重启服务器, TAB补全才能生效)
#
RegisterCommands:
  # 主命令
  openMenus:
    # 别称
    aliases: []
    # 权限
    permission: null
    # 无参数执行时
    execute:
      - 'tell: &7Argument Required!'
    # 参数及对应的反应
    arguments:
      example: 'open: example'

#
# 控制一些插件监听的 Bukkit 事件，是否忽略已被取消的事件
# 如果遇到一些兼容性问题可以尝试调整
#
Events-Ignore-Cancelled:
  # 监听菜单打开命令
  PlayerCommandPreprocessEvent: true
```

迁移

支持插件

- TrMenu v1.x

迁移方法

- 执行子命令 Migrate
- /TrMenu Migrate TrMenuV1 [File/Dir]

MENU

结构

菜单

- 标题
  - 单个或多个标题
  - 标题更新周期
- 布局
  - 菜单布局
  - 玩家容器布局
- 选项
  - 默认补全参数
  - 默认布局页码
  - 是否隐藏玩家容器物品
  - 防频繁点击间隔
  - 需要依赖的 PlaceholderAPI 拓展
- 绑定
  - 绑定正则命令
  - 绑定物品特征
- 事件
  - 开启菜单执行动作
  - 关闭菜单执行动作
- 图标
- 内置脚本
- 周期任务

标题

示例

```yaml
#
# 菜单的标题，支持动态标题周期切换
#
Title:
  - 'Hello, TrMenu!'
  - 'Support Animated Titles'

#
# 若有多个标题，切换的周期（ticks）
#
Title-Update: 40
```

注意

- **标题更新周期** 只有当存在多个标题时有效
- 若使用非动态标题，可以使用 **设置标题** 等动作对标题二次修改

布局

示例

```yaml
#
# 菜单的布局功能
# 你可以用 ` ` 来标记设置多个字符名称图标的位置
#
Layout:
  - '########`Close`'
  - '         '
  - '         '
  - '         '
  - '########`Next`'

#
# 玩家容器菜单的布局
#
PlayerInventory:
  - '         '
  - '         '
  - '         '
  - '         '
```

- 每个字符代表一个图标在菜单中的位置
- 布局的行数将同时定义箱子容器的大小
- 你也可以用 `` 将字符串包裹起来，从而不受限于单个字符
- 玩家容器菜单布局仅限 4*9 大小

多页

```yaml
Layout:
  - - '########`Close`'
    - '         '
    - '         '
    - '         '
    - '########`Next`'
```

```
7
8     - - '########`Close`'
9       - '         '
10      - ' *       '
11      - '         '
12      - '`Pre`########'
```

- 布局功能可以支持多个布局，从而在同一个菜单中快速实现多页功能

注意

- 若无 **玩家容器菜单布局** 与 **菜单布局** 对应，则在该菜单页玩家容器为空
- 标识图标的字符支持汉字

选项

示例

```yaml
#
# 菜单的选项设置
#
Options:
  # 是否启用菜单传参功能 （默认开启）
  Arguments: true
  # 默认填充参数
  Default-Arguments: []
  # 默认布局页码
  Default-Layout: 0
  # 是否隐藏玩家容器
  Hide-Player-Inventory: true
  # 防频繁点击的间隔
  Min-Click-Delay: 200
  # 强制需要依赖的 PlaceholderAPI 拓展变量
  Depend-Expansions: ['server', 'player', 'progress']
```

注意

- 若设置 **默认填充参数** ，则将自动补全玩家未提供的参数，例如
  - Default-Arguments: ["TabooLib", "Virus"]
  - 玩家提供的参数为 ["TabooLib"], 则第二个参数将自动补全为 Virus
- **默认菜单布局页码** 即在未指定页码的情况下默认打开菜单后初次显示的页码，在 Open 命令中也可以指定页码

绑定

示例

```yaml
#
# 菜单绑定的快捷打开方式
#
Bindings:
  # 绑定命令，支持正则
  Commands:
    - 'tester'
  # 绑定到物品特征
  Items:
    - 'material:compass'
```

注意

- 绑定命令支持**正则匹配**，例如添加 (?i) 前缀可以忽略大小写
- 绑定命令支持空格 & 其它参数，插件将自动匹配并读取玩家提供的真实参数
- 了解绑定 **物品特征** 的详细写法，请查看下面章节

→ 物品特征 /functions/item-identifiers

事件

示例

```yaml
#
# 菜单的事件处理
#
Events:
  # 开启菜单事件
  Open:
    - requirement: 'hasPerm.trmenu.use or is.{reason}.RELOAD or is.{reason
      actions:
        - 'sound: BLOCK_CHEST_OPEN-1-0'
      deny-actions:
        - 'sound: ENTITY_ITEM_BREAK-1-0'
        - 'title: <title=&c&l权限不足><subtitle=&7&l本菜单需要权限 &6&ltrmenu
        - 'return'
  # 关闭菜单事件
  Close:
    - 'sound: BLOCK_CHEST_CLOSE-1-0'
```

注意

- 事件中的反应写法后面会详解
- 在开启菜单事件中，{reason} 变量将返回

```
me.arasple.mc.trmenu.api.events.MenuOpenEvent.Reason
```

- 若执行 return 动作，事件将被取消

图标

示例

```yaml
#
# 菜单的图标
#
Icons:
  # 图标的唯一 ID，可配合布局使用
  '#':
    # 显示图标的动态更新周期，若设置多个则依次对应为 Material，Name，Lore，Slots
    # 设置一个即为所有项相同更新周期
    update: [-1, 10, 15, -1]
    # 菜单的显示部分
    display:
      material: 'Gray Stained Glass Pane'
      name: ['&fTr&7Menu', '&7Tr&8Menu', '&8Tr&0Menu','&7Tr&8Menu']
      lore:
        - - '&7Thanks &f:> &7for using!'
        - - '&7Thanks &f:) &7for using!'
    # 点击菜单后执行的动作
    actions:
      all: 'sound: BLOCK_NOTE_BLOCK_PLING-1-2'
```

```yaml
#
# 优先级图标示例
#
Icons:
  '#':
    update: 20
    display:
      material: 'Gray Stained Glass Pane'
      name: '&7Normal'
    actions:
      all: 'sound: BLOCK_NOTE_BLOCK_PLING-1-2'
    icons:
      # 条件，拥有 mvp.user
      - condition: 'hasPerm.mvp.user'
        prioriry: 2
        # 深度继承默认图标
        inherit: true
        display:
          name: '&bMVP'
```

```
      actions:
        all: 'tell: Hello, MVP User!'
    # 条件，拥有 vip.user
    - condition: 'hasPerm.vip.user'
      priority: 1
      display:
        name: '&aVIP'
      actions:
        all: 'tell: Hello, VIP User!'
```

结构

- 更新周期
  - 材质
  - 名称
  - Lore
  - 槽位
- 刷新周期（重新计算优先级）
- 显示部分
- 动作部分
- 子图标
  - 条件
  - 优先级
  - 是否继承
  - 显示部分
  - 动作部分

注意

- 所有图标均配置在 **Icons** 节点下
- 更新周期支持多个属性（材质，名称，Lore，槽位）的独立周期，若只配置一个则默认全部（如 v1）
- 子图标优先级按降序，选取第一个符合条件的子图标

- 默认情况下，子图标仅继承默认图标的位置、材质，若开启继承且未设置名称，Lore，则将继承
- 材质、名称、Lore、槽位等均支持动态效果

→ 图标显示 /menu/icons/display

内置脚本

示例

无参数脚本

```
#
# 菜单内置的自定义脚本变量
# 通过 ${ID_参数_参数} 的方式可以调用
#

Functions:
  health: |-
    function math(){
      return player.getHealth()
    }
    math()
```

带参数脚本

```
Functions:
  flash: |-
    function flash() {
      var parsed = "%animations_<flash>{0}</flash>%"
      return parsed.isEmpty() ? utils.emptyString("{0}".length()) : parsed
    }
    flash()
```

调用

- 每个内置脚本都有独一无二的 Id 标识
- 在菜单内容中，你可以使用 **${Id}** 的形式调用
- 若需提供参数，则按照 **${Id_参数1_参数2}** 的格式调用
- 参数在脚本中以 {0}, {1} ... 的形式使用，若需要菜单传参，则使用 PlaceholderAPI 变量

```
```

```yaml
'Health':
  update: 20
  display:
    material: 'Red Stained Glass Pane'
    name: 'Health'
    lore:
    - ''
    - '&cHealth: ${health}'
    - ''
  actions:
    all: 'close'
```

```yaml
# ...
lore:
 - '&b${flash_➥} &3Left-Click &7to display more info.'
```

周期任务

示例

```yaml
#
# 菜单自定义周期性任务
#

Tasks:
  # 任务 Id
  tikTok:
    # 周期 ( Ticks )
    period: 20
    # 反应
    task:
      - requirement: 'isOperator.'
        actions:
          - 'sound: BLOCK_NOTE_BLOCK_BIT-1-2'
```

结构

- 每个周期任务由 **任务 Id**、**周期**、**反应** 三部分组成
- 玩家开启菜单后，所有周期任务会启动，关闭菜单后会停止

→ 反应 /actions/reactions

图标

更新周期

解释

- 更新周期包括 4 个（材质，名称，Lore，槽位）
- 你可以通过配置数组的形式独立设置它们

示例

```yaml
# 设置一个值，同时应用到 材质，名称，Lore，槽位 中
update: 20
```

```yaml
# 不完全设置
# 分别设置材质，名称，Lore的更新周期为 20, -1, 15
# 槽位的更新周期将选取其中最高的一个，即 20
update: [20, -1, 15]
```

```yaml
# 完全配置，即按顺序分别对应
update: [-1, 5, 25, 30]
```

注意

- 不配置更新周期项，图标将默认不更新
- 更新周期应与默认图标同级节点，子图标不支持自定义更新周期

刷新周期

示例

```
refresh: 20
```

注意

- 只有存在子图标时，此项才有效
- 使用动作 Refresh，可以主动刷新图标，达到同样效果

图标显示

材质

节点

```
(mat(erial)?|texture)(s)?
```

示例

```
'*':
  update: [-1, 5, 20, -1]
  display:
    material: '<skull:9842dc3b917b1a796c303e15105474a8e315de7982b6ca54fe
```

```
'Close':
  update: [25, 5, -1, -1]
  display:
    # 动态材质，更新周期 25 ticks
    materials:
    - 'Red Stained Glass Pane'
    - 'Orange Stained Glass Pane'
    name: ['&cC&7lose', '&cCl&7ose', '&cClo&7se', '&cClos&7e', '&cClose'
```

原版物品

```
# 直接书写标准材质名称
material: 'CYAN_STAINED_GLASS_PANE'

# 忽略大小写
material: 'Cyan Stained Glass Pane'
```

```
# (1.8-1.12) 数字 Id
material: 160

# (1.8-1.12) 数字 Id + 子 Id
material: '3:2'
```

Data 值

```
# 同上 '3:2'
material: '3<data:2>'
```

CustomModelData 值

```
# (1.14+) <ModelData> 标签前写具体的物品材质
material: 'coal<model-data:1>'
```

玩家头颅

```
# 直接点名
material: '<head:BlackSKY>'

# 通过变量获取
material: '<head:%player_name%>'
```

- TrMenu 采用本地 NMS 读取在线玩家 + 异步联网读取离线玩家 皮肤材质
- 不会卡顿服务器，放心使用

纹理头颅

```
# 类型 I
material: '<skull:eyJ0ZXh0dXJlcyI6eyJTS0lOIjp7InVybCI6Imh0dHA6Ly90ZXh0dXJl
# 类型 II
material: '<skull:44f452d998eabac4642c6b0fe5a8f4e2e673edcae2a6dfd9e6a2e86e
```

Custom heads https://minecraft-heads.com/custom-heads

染色皮革

```
# <Dye> 标签前面写皮革物品对应的原版材质名
material: 'leather chestplate<dye:255,255,0>'
```

自定义旗帜

```
material: 'white banner<banner:RED MOJANG,WHITE>'
```

PatternType (Spigot-API 1.16.5-R0.1-SNAPSHOT API) https://hub.spigotmc.org/javadocs/bukkit/org/bukkit/block/banner/PatternType.html

JSON 物品

```
material: '{"type":"DIAMOND_SWORD","data":0,"amount":1,"meta":{"Damage":{"ty
```

- 通过 Item 子命令将物品转为 JSON 文本格式, 直接粘贴使用
- 支持一切物品材质

物品仓库

```
material: '<repo:My_Custom_Item>'
```

HeadDatabase 物品

```
# 头颅 Id
material: '<head-database:100>'

# 随机头颅
material: '<head-database:random>'
```

SkinsRestorer 头颅

```
material: '<skins-restorer:JYBH>'
```

- 对于安装了 SkinsRestorer 皮肤插件的离线服务器，可以使用此类型来取得玩家头颅
- 同理支持使用变量

Oraxen 物品

```
material: '<oraxen:Id>'
```

名称

节点

```
(display)?(-)?name(s)?
```

示例

```yaml
'Next':
  display:
    material: 'Cyan Stained Glass Pane'
    name: '&bN&3ext Page'
```

```yaml
'Close':
  update: [25, 5, -1, -1]
  display:
    materials:
    - 'Red Stained Glass Pane'
    - 'Orange Stained Glass Pane'
    # 动态名称，更新周期 5 ticks
    name: ['&cC&7lose', '&cCl&7ose', '&cClo&7se', '&cClos&7e', '&cClose'
```

Lore

节点

```
(display)?(-)?lore(s)?
```

示例

```yaml
# 嵌套 List，多组动态 Lore
lore:
  - - '&7Thanks &f:> &7for using!'
  - - '&7Thanks &f:) &7for using!'

# 静态 Lore 描述
lore:
  - 'Hello There'
  - 'Hello TrMenu!'
```

条件

> 此功能将在每次更新 Lore 时进行计算条件，可能影响插件性能表现

```yaml
lore:
  - 'Line 1'
  - 'Line 2'
  # 当玩家有 vip.user 权限时显示此行
  - 'You are VIP User<condition: hasPerm.vip.user>'
  - 'Line 3'
```

小技巧

- Lore 中支持使用 \n 换行，也支持变量中的换行符号
- 可以在配置文件中配置默认 Lore 行的颜色

数量

节点

```
(amt|amount)(s)?
```

示例

```yaml
# 静态数量
amount: 10

# 动态数量（需要是返回数值的变量）
# v2 版中此项不再默认进行 Js 运算
amount: '%server_time_s%'
```

注意

- 物品的动态数量不提供独立的更新周期设置，将随着每次物品刷新更新

发光

节点

```
(shiny|glow)(s)?
```

示例

```yaml
# 直接设置布尔值，开启或关闭
shiny: true

# 动态效果（条件表达式）
shiny: 'hasPerm.vip.user and hasMoney.1000'
```

误用示范

繁琐的条件图标

```yaml
A:
  refresh: 20
  update: 20
  display: # ...
  actions: # ...
  icons:
  - condition: 'hasPerm.vip.user'
    display:
      glow: true
```

简洁一步到位

```yaml
A:
  update: 20
  display:
    # ...
    shiny: 'hasPerm.vip.user'
  actions: # ...
```

注意

- 物品的发光效果不提供独立的更新周期设置，将随着每次物品刷新更新

标签

节点

```
flag(s)?
```

示例

```yaml
flags:
- 'HIDE_ENCHANTS'
- 'HIDE_ATTRIBUTES'
```

可用标签

- HIDE_ENCHANTS
- HIDE_ATTRIBUTES
- HIDE_UNBREAKABLE
- HIDE_POTION_EFFECTS
- HIDE_DESTROYS
- HIDE_PLACED_ON

NBT

节点

```
nbt(s)?
```

示例

```yaml
nbt:
  balabala: 12345
```

- 支持使用变量

动作交互

示例

```yaml
'Close':
  update: [-1, 5, -1, -1]
  display:
    material: Red Stained Glass Pane
    name: ['&cC&7lose', '&cCl&7ose', '&cClo&7se', '&cClos&7e', '&cClose']
  # 动作部分
  actions:
    # 点击类型 - 反应
    all: close
```

```yaml
  '*':
    update: [-1, 5, 20, -1]
    display: # ...
    # 动作部分
    actions:
      # 点击类型（左键）
      left:
        - 'set-meta: icon_server_hide true'
        - 'sound: BLOCK_NOTE_BLOCK_BIT-1-0'
        - 'refresh'
```

点击类型

```java
package me.arasple.mc.trmenu.api.inventory

/**
 * @author Arasple
 * @date 2020/3/22 10:23
 */
enum class InvClickType {

    /**
```

```
10          * 左键点击
11          */
12         LEFT,
13
14         /**
15          * 按住 Shift 并左键点击
16          */
17         SHIFT_LEFT,
18
19         /**
20          * 右键点击
21          */
22         RIGHT,
23
24         /**
25          * 按住 Shift 并右键点击
26          */
27         SHIFT_RIGHT,
28
29         /**
30          * 中键点击（摁住鼠标滚轮）
31          */
32         MIDDLE,
33
34         /**
35          * 按住 Shift 并中键点击
36          */
37         DOUBLE_CLICK,
38
39         /**
40          * 丢弃键（默认为 Q）
41          */
42         DROP,
43
44         /**
45          * 切换副手（默认为 F）（1.16+）
46          */
47         OFFHAND,
48
49         /**
50          * Ctrl + 丢弃键
51          */
52         CONTROL_DROP,
53
54         /**
55          * 创造模式 + 中键
56          */
57         CREATIVE,
58
59         /**
60          * 数字键 1 - 9
```

```
61          */
62         NUMBER_KEY,
63
64         /**
65          * 数字键 1
66          */
67         NUMBER_KEY_1,
68
69         /**
70          * 数字键 2
71          */
72         NUMBER_KEY_2,
73
74         /**
75          * 数字键 3
76          */
77         NUMBER_KEY_3,
78
79         /**
80          * 数字键 4
81          */
82         NUMBER_KEY_4,
83
84         /**
85          * 数字键 5
86          */
87         NUMBER_KEY_5,
88
89         /**
90          * 数字键 6
91          */
92         NUMBER_KEY_6,
93
94         /**
95          * 数字键 7
96          */
97         NUMBER_KEY_7,
98
99         /**
100         * 数字键 8
101         */
102        NUMBER_KEY_8,
103
104        /**
105         * 数字键 9
106         */
107        NUMBER_KEY_9,
108
109        /**
110         * 点击容器左侧区域
111         */
```

```
112         WINDOW_BORDER_LEFT,
113
114         /**
115          * 点击容器右侧区域
116          */
117         WINDOW_BORDER_RIGHT,
118
119         /**
120          * 所有触发方式
121          */
122         ALL;
123
124     }
```

小技巧

- 你可以为同一个反应快速添加多种触发的点击方式，用逗号分隔开，例如

```
1  actions:
2    # 左键、中键 或按 数字键 8
3    left,middle,number_key_8:
4     - 'tell: Hello'
```

注意

- **反应** 支持多种写法，与 周期任务、菜单事件 中的相同，后面会详解
- 动作部分节点与显示部分节点同级
- 默认图标和子图标都可以有独立的动作交互部分

优先级

示例

```
priority: 20
```

注意

- 该项仅对子图标有效
- 若不设置，则默认优先级为 **-1**
- 选取条件子图标时，按优先级**降序**依次计算子图标到满足条件的为止

> 当图标仅有 **一个优先级图标** 需求时，可以不配置优先级

条件

示例

```
condition: hasPerm.mvp.user
```

注意

- 条件表达式写法后面会详解

→ 表达式 /script/expressions

继承

示例

```
inherit: true
```

注意

- 默认情况下，子图标将继承默认图标的材质、槽位
- 开启继承，且子图标未设置名称或Lore，则将从默认图标中继承

ACTIONS

动作

介绍

- TrMenu 提供大量完善的 **动作** ，皆在提高菜单制作效率 & 减少繁琐命令
- 无论是显示 Title、Actionbar，还是构建一条支持变量的 Json 消息，你都可以轻松实现
- 除此之外，每条动作都支持多个独立的动作 **选项** ，包括延时、条件、概率和遍历等

变量

- 动作的所有内容均支持使用 **PlaceholderAPI 变量，颜色代码**
- 也支持使用 TrMenu 的 菜单参数, Meta 元素，内置函数 等

捆绑

- 将多个动作绑定到一行内容，动作选项共享
- 使用 **&&&** 或 **_**//\ 连接** ，**例如_

```
- 'rem-meta: angryBee &&& refresh &&& sound: ENTITY_SPIDER_STEP-1-2<delay:

# 实质上等同于
- 'rem-meta: angryBee<delay:80>'
- 'refresh<delay:80>'
- 'sound: ENTITY_SPIDER_STEP-1-2<delay:80>'
```

执行命令

节点

```
command|cmd|player|execute
```

示例

```
- 'command: spawn'
```

```
# 以 ; 分隔，可以执行多个命令
# 下方动作等同于
# - 'command: spawn'
# - 'command: say I am Back'

- 'command: spawn ; say I am Back'
```

执行命令 (OP)

节点

```
op(erator)?(s)?
```

示例

```
- 'op: kick ${input}'
```

```
- 'op: kick ${input} ; broadcast I just kicked ${input}'
```

注意

> ⚠ 当前情况下，Bukkit 所能实现此效果的方法只能是 **授予 OP，执行操作，撤销 OP**
>
> 在可以用 **CONSOLE** 控制台执行命令实现需求的情况下，**不推荐**使用此动作

执行命令 (CONSOLE)

节点

```
console
```

示例

```
- 'console: lp perm add %player_name% vip.user ;give %player_name% diamond 6
```

执行发送

节点

```
chat
```

示例

```
# 支持 ; 分隔多条内容
- 'chat: I Love TabooLib :);For real'
# 支持模拟执行命令
- 'chat: /quit'
```

发送消息

节点

```
tell|send|message|msg|talk|say
```

示例

```
- 'tell: Hello There'
```

```
- 'tell: Hello There;There is a message'
- 'tell: First Line\nSecond Line!'
```

发送 Tellraw

节点

```
tellraw|json
```

示例

原版 JSON

```
- 'json: {"text":"Hello World!"}'
```

构建消息 1

```
- 'json: Hello World! <ClickMe?command=spawn&hover=to spawn> <--- Clic
```

构建消息 2

```
- 'json: &3Hello, &b{player_name}&3, Buy Ranks on our <&2&nstore?url=h
```

发送 Actionbar

节点

```
action(bar)?(s)?
```

示例

```
- 'actionbar: Hello There'
```

发送 Title & SubTitle

节点

```
(send)?(-)?(sub)?title(s)?
```

示例

```
# 旧版写法，需要以 <Key=Value> 的格式指定各属性
- 'title: <TITLE=WELCOME><SUBTITLE=%player_name%~><FADEIN=20><STAY=20><FAD
```

```
# 新版简写方法
# 用 \s 代替标题中需要使用的空格
- 'title: WELCOME %player_name%~ 20 20 10'
```

音效

节点

```
(play)?(-)?sound
```

示例

```
- 'sound: BLOCK_ANVIL_HIT-1-2'
```

```
- 'sound: BLOCK_BEEHIVE_ENTER-1-2;BLOCK_BUBBLE_COLUMN_WHIRLPOOL_INSIDE-1-0'
```

跨服传送

节点

```
bungee|server|connect
```

示例

```
- 'server: Lobby'
```

执行脚本

节点

```
(java)?(-)?script(s)?|js
```

示例

```
- 'js: player.setNoDamageTicks(600)'
- 'js: player.setVelocity(player.getVelocity().setY(25))'
- 'js: bukkitServer.broadcastMessage("A Broadcast Message")'
```

了解更多

→ 脚本 /script/scripts

输入捕获

节点

```
(input)?(-)?catcher(s)?
```

旧版写法示例

```
- |-
  Catcher:
   <Type=CHAT>
   <Before=tell:请输入一个值>
   <Valid=tell: 你成功输入了一个数值 ${input}>
   <Invalid=tell: 你输入的不是一个数值>
   <Require=isNumber.${input}>
   <Cancel=tell: 已取消捕获器>
```

Type

捕获器的类型

Before

玩家输入内容之前，执行的动作

Valid

玩家输入值，并且满足 Require 条件时执行的动作

Invalid

玩家输入值，并且不满足 Require 条件时执行的动作

Require

玩家输值后，满足此条件则执行 Valid 动作，否则执行 Invalid 中动作

高级写法示例

```
- catcher:

    target:
      type: CHAT
      before: 'tell: 请输入一名玩家的名称'
      cancel: 'tell: 已取消捕获器'
      reactions:
        - condition: 'isOnline.${input_target}'
          execute: 'tell: 名称确认. ${input_target}'
          deny:
            - 'tell: 抱歉, 但玩家 ${input_target} 不在线'
            - 'return'

    amount:
      type: CHAT
      before: 'tell: 请提供支付数额'
      cancel: 'tell: 已取消捕获器'
      reactions:
```

```
19          - condition: 'isNumber.${input_amount}'
20            execute:
21              - 'tell: 数额确认. ${input_amount}'
22              - 'command: pay ${input_target} ${input_amount}'
23            deny:
24            - 'tell: 请提供有效数字'
25            - 'retype'
```

- 执行到中断动作 **Return** 将会中断该输入捕获器
- 执行到重新输入动作 **Retype** 将重新执行当前的子捕获器

→ 重新输入 /actions/actions/input-catcher/reinput

类型

- CHAT
  - 捕获玩家发送到聊天框的内容
- SIGN
  - 向玩家发送一个虚拟木牌，捕获输入的内容
- ANVIL
  - 向玩家发送一个虚拟铁砧，捕获命名的内容

重新输入

节点

```
re(-)?(catcher|input|enter|type)(s)?
```

示例

```
- 'retype'
```

辅助

延时

节点

```
delay|wait
```

示例

```
- 'tell: The diamond reward will be distributed in 3 seconds'
- 'delay: 60'
- 'give-item: material:DIAMOND'
```

中断

节点

```
return|break|cancel
```

物品

扣除物品

节点

```
(take|remove)(-)?item(s)?
```

示例

```
# 扣除一组钻石
- 'take-item: material:DIAMOND,amount:64'
```

```
# 扣除一组铁锭和半组金锭
- 'take-item: material:IRON_INGOT,amount:64;material:GOLD_INGOT,amount:32'
```

注意

- 扣除物品的动作并**不会主动检测**玩家背包是否包含该物品，实现商店需要配合条件表达式
- 物品特征写法和其它部分完全统一

→ 物品特征 /functions/item-identifiers

→ **hasItem** /script/expressions/hasitem

给予物品

节点

```
(give|add)(-)?item(s)?
```

示例

```
# 给予半组显示名称为 "&bCustom Diamond" 的钻石
- 'give-item: material:DIAMOND,amount:32,name:&bCustom Diamond'
```

注意

- 支持给予通过命令转换的 JSON 格式物品

附魔物品

节点

```
enchant(-)?item(s)?
```

修复物品

节点

```
repair(-)?item(s)?
```

经济

给予金币

节点

```
(give|add|deposit)(-)?(money|eco|coin)(s)?
```

示例

```yaml
- 'give-money: %server_time_s%'
- 'give-money: 100'
```

扣除金币

节点

```
(take|remove|withdraw)(-)?(money|eco|coin)(s)?
```

示例

```
- 'take-money: 100'
```

设置金币

节点

```
set(-)?(money|eco|coin)(s)?
```

示例

```
- 'set-money: 0'
```

给予点卷

节点

```
(give|add|deposit)(-)?point(s)?
```

示例

```
- 'give-points: 100'
```

扣除点卷

节点

```
(take|remove|withdraw)(-)?point(s)?
```

示例

```
- 'take-points: 100'
```

设置点卷

节点

```
set(-)?point(s)?
```

示例

```
- 'set-points: 0'
```

菜单

关闭菜单

节点

```
close
```

示例

```
- 'close'
```

关闭菜单 (静默)

节点

```
(force|silent)(-)?close
```

示例

```
- 'silent-close'
```

打开菜单

节点

```
open(s)?|(open)?(-)?gui|(tr)?menu
```

示例

```
- 'open: Example'
```

```
# 打开菜单 Shop-Handler,
# 指定参数为 [APPLE, 10, 1]

- 'open: Shop-Handler APPLE 10 1'
```

```
# 打开菜单 Test 的页码 3,
# 指定参数为 [Hello Arasple, Arg2]

- 'open: Test:2 `Hello Arasple` Arg2'
```

切换页码

节点

```
((set|switch)?(-)?(shape|page))|shape|page
```

示例

```
- 'page: 0'
```

注意

- 页码从 0 开始，例如 0 对应第一页，1 对应第二页
- 你可以使用变量

设置参数

节点

```
set(-)?arg(ument)?(s)?
```

示例

```
- 'set-arguments: `Custom Argument` 10 Hello'
```

设置标题

节点

```
set(-)?title
```

示例

```
- 'set-title: CUSTOM_TITLE'
```

刷新图标

节点

```
(icon)?(-)?(refresh|update)
```

示例

```
- 'refresh'
```

数据

介绍

- TrMenu 目前提供 Meta / Data 两种变量数据变量
- 前者（Meta）存储在内存中，将在插件重载后失效
- 后者（Data）玩家独立存储在本地 SQLite 中，不会失效

应用

- 设置 Meta 或 Data 后，你都可以以变量的形式调用它们

```
{meta:<KeyName>}
```

```
{data:<KeyName>}
```

示例

```yaml
Bee:
  update: [-1, -1, -1, 20]
  display:
    material: '<skull:b727d0ab03f5cd022f8705d3f7f133ca4920eae8e1e47b507443
    name: '&eBee ~'
    slots:
      - [9]
      - [18]
      - [28]
      - [29]
      - [30]
      - [31]
```

```
      - [32]
      - [33]
      - [34]
      - [35]
      - [26]
      - [16]
      - [15]
      - [14]
      - [13]
      - [12]
      - [11]
      - [10]
  actions:
    all:
      - 'set-meta: angryBee true'
      - 'sound: ENTITY_ENDERMAN_HURT-1-2'
      - 'rem-meta: angryBee &&& refresh &&& sound: ENTITY_SPIDER_STEP-1-2<
      - 'refresh'
  icons:
    - condition: 'equals.{meta:angryBee}.true'
      inherit: true
      display:
        name: '&cBee &4~'
        material: '<skull:e6b74e052b74288799ba6d9f35c5d0221cf8b04331547ec2
      actions:
        all: 'sound: ENTITY_ENDERMAN_HURT-1-2'
```

设置 Meta

节点

```
set(-)?(temp|var(iable)?|meta)(s)?
```

示例

```
- 'set-meta: Key1 Value1'
- 'set-meta: Key2 Value2;Key3 Value3'
```

删除 Meta

节点

```
(remove|rem|del)(-)?(temp|var(iable)?|meta)(s)?
```

示例

```
- 'rem-meta: Key1'
- 'rem-meta: Key2;Key3'
```

设置 Data

节点

```
(set|edit)(-)?(data)(s)?
```

示例

```
- 'set-data: Key1 Value1'
- 'set-data: Key2 Value2;Key3 Value3'
```

删除 Data

节点

```
(remove|rem|del)(-)?(data)(s)?
```

示例

```
- 'del-data: Key1'
- 'del-data: Key2;Key3'
```

其他

Cronus Effect

节点

```
(cq|cronus)(-)?effect(s)?
```

TabooLib Hologram

TabooLib Particle

选项

格式

```
<OptionName=OptionValue>
```

延时执行

节点

```
<(d|delay|wait)[:=]( )?([0-9]+[.]?[0-9]*>)
```

示例

```
- 'tell: Delayed Message<Delay=20>'
```

执行条件

节点

```
<(r|require(ment)?|condition)[:=]( )?(.+>)
```

示例

```
- 'tell: VIP Message<Condition=hasPerm.user.vip>'
```

→ 表达式 /script/expressions

执行概率

节点

```
<(c|chance|rate)[:=]( )?([0-9]+[.]?[0-9]*>)
```

示例

```
- 'give-item: material:DIAMOND<Chance=0.8>'
```

遍历执行

节点

```
<((p|(for|all)?(-)?players))[:=]?(.+)?>
```

示例

```
1  - 'tell: broadcast message<Players>'
2  - 'tell: broadcast message for VIP<Players:hasPerm.vip.user>'
```

反应

常规写法

```
actions:
 - 'tell: Hello, %player_name%'
```

三元写法

```
actions:
  condition: 'hasMoney.100'
  actions:
  - 'tell: You have enough money.'
  deny-actions:
  - 'tell: You don''t have enough money.'
```

多组三元写法

```
    actions:
      all:
        - condition: 'utils.chance(0.1)'
          priority: 3
          actions:
            - 'tell: Prize 1'
            - 'return'
        - condition: 'utils.chance(0.5)'
          priority: 2
          actions:
            - 'tell: Prize 2'
            - 'return'
        - condition: 'true'
          priority: 1
```

```
15            actions:
16              - 'tell: Prize 3'
```

注意

- 使用多个反应组时，可以设置优先级，按降序执行
- 若执行到任意**中断动作**，整个动作组将停止
- 节点别称有许多，例如

```
1  requirement: 'hasPoints.100'
2  execute: 'tell: You have 100 points'
3  deny: 'tell: You dont have 100 points'
```

SCRIPT

表达式

- TrMenu v2 提供全新的表达式写法，简单易懂
- 暂时不支持自定义添加
- 参数通常由 . 分开

示例

```
# 同时拥有权限 test 和 等级 >= 10
hasPerm.test and hasLevel.10
```

连词

- and = &&
  - "与"，需要相连的两部分均返回 true 时，表达式才返回 true
- or = ||
  - "或者"，需相连两部分任意一部分返回 true 时，表达式即返回 true

stringEquals

示例

```
equals.%player_name%.Arasple
```

```
is.{meta:hide}.true
```

stringEqualsIgnoreCase

示例

```
equalsIgnoreCase.%player_name%.Arasple
```

isNumber

示例

```
isNumber.${input}
```

isOperator

示例

```
isOperator.
```

isPlayerOperator

示例

```
isPlayerOperator.BlackSKY
```

hasPermission

示例

```
hasPermission.userData.test
```

```
hasPerm.vip.user
```

hasLevel

示例

```
hasLevel.10
```

hasMoney

示例

```
hasMoney.100
```

```
hasMoney.{math_${input_amount}*100}
```

hasPoints

示例

```
hasPoints.100
```

hasItem

示例

```
hasItem.material:DIAMOND,amount:64
```

物品特征 /functions/item-identifiers

脚本

对象

Name	Description
player	玩家对象 org.bukkit.entity.Player
bukkitServer	Bukkit 服务器对象 Bukkit.getServer()
utils	辅助工具类 me.arasple.mc.trmenu.utils.Assist

Assist Utils

```
package me.arasple.mc.trmenu.utils

import io.izzel.taboolib.internal.apache.lang3.math.NumberUtils
import io.izzel.taboolib.module.compat.EconomyHook
import io.izzel.taboolib.module.tellraw.TellrawJson
import io.izzel.taboolib.util.item.ItemBuilder
import io.izzel.taboolib.util.item.Items
import io.izzel.taboolib.util.lite.Numbers
import me.arasple.mc.trmenu.data.MetaPlayer.getArguments
import me.arasple.mc.trmenu.modules.action.Actions
import me.arasple.mc.trmenu.modules.hook.HookCronus
import me.arasple.mc.trmenu.modules.hook.HookPlayerPoints
import me.arasple.mc.trmenu.modules.item.ItemIdentifierHandler
import me.arasple.mc.trmenu.modules.web.WebData
import me.clip.placeholderapi.PlaceholderAPI
import org.bukkit.*
import org.bukkit.entity.Player
import org.bukkit.inventory.ItemStack


/**
 * @author Arasple
 * @date 2020/7/21 20:57
 */
class Assist {

```

```kotlin
    fun runAction(player: Player, vararg actions: String) = actions.filter

    fun parseBracketPlaceholders(player: OfflinePlayer, string: String): S

    fun connect(player: Player, server: String) = Bungees.connect(player,

    fun sendBungeeData(player: Player, vararg data: String) = Bungees.send

    fun getPlayerArgs(player: String): Array<String> = getPlayer(player)?.

    fun isPlayerOperator(player: String) = getPlayer(player).let { return@

    fun isPlayerOnline(player: String) = getPlayer(player).let { return@le

    fun getPlayer(player: String): Player? = Bukkit.getPlayerExact(player)

    fun getOnlinePlayers(): Collection<Player> = Bukkit.getOnlinePlayers()

    @ExperimentalStdlibApi
    fun getRandomPlayer(): Player? = Bukkit.getOnlinePlayers().randomOrNul

    fun getItemBuildr(): ItemBuilder = ItemBuilder(Material.STONE)

    fun getTellraw(): TellrawJson = TellrawJson.create()

    fun emptyString(length: Int) = buildString {
        for (i in 0..length) append(" ")
    }

    fun equalsIgnoreCase(sample: String, temp: String) = sample.equals(tem

    fun chance(number: String) = Numbers.random(NumberUtils.toDouble(numbe

    fun randomInteger(low: Int, high: Int): Int = IntRange(low, high).rand

    fun randomDouble(low: Double, high: Double) = Numbers.getRandomDouble(

    fun isNumber(number: String) = NumberUtils.isParsable(number)

    fun isInt(number: String) =
        try {
            number.toInt()
            true
        } catch (e: Throwable) {
            false
        }


    fun toInt(number: String) = NumberUtils.toInt(number, -1)

    fun toDouble(number: String) = NumberUtils.toDouble(number, -1.0)
```

```
78
79      fun isWithin(input: String, low: String, high: String) = IntRange(low.
80
81      fun isSmaller(input1: String, input2: String) = NumberUtils.toDouble(i
82
83      fun isSmallerOrEqual(input1: String, input2: String) = NumberUtils.toD
84
85      fun isGreater(input1: String?, input2: String?) = NumberUtils.toDouble
86
87      fun isGreaterOrEqual(input1: String?, input2: String?) = NumberUtils.t
88
89      fun createLocation(world: String?, x: Double, z: Double): Location? {
90          val y = Bukkit.getWorld(world!!)!!.getHighestBlockYAt(x.toInt(), z
91          return createLocation(world, x, y, z)
92      }
93
94      fun createLocation(world: String?, x: Double, y: Double, z: Double): L
95          return createLocation(world, x, y, z, 0f, 0f)
96      }
97
98      fun createLocation(world: String?, x: Double, y: Double, z: Double, ya
99          return Location(Bukkit.getWorld(world!!), x, y, z, yaw, pitch)
100     }
101
102     fun createLocation(world: World, x: Double, z: Double): Location? {
103         val y = world.getHighestBlockYAt(x.toInt(), z.toInt()).toDouble()
104         return createLocation(world, x, y, z)
105     }
106
107     fun createLocation(world: World?, x: Double, y: Double, z: Double): Lo
108         return createLocation(world, x, y, z, 0f, 0f)
109     }
110
111     fun createLocation(world: World?, x: Double, y: Double, z: Double, yaw
112         return Location(world, x, y, z, yaw, pitch)
113     }
114
115     fun hasMoney(player: Player, money: String) = hasMoney(player, NumberU
116
117     fun hasMoney(player: Player, money: Double) = EconomyHook.get(player)
118
119     fun hasPoints(player: Player, points: String) = hasPoints(player, Numb
120
121     fun hasPoints(player: Player, points: Int) = HookPlayerPoints.hasPoint
122
123     fun getItemName(itemStack: ItemStack): String = Items.getName(itemStac
124
125     fun query(url: String) = WebData.query(url)
126
127     fun hasItem(player: String, identify: String) = getPlayer(player)?.let
128
```

```
129     fun hasItem(player: Player, identify: String) = ItemIdentifierHandler.
130
131     fun evalCronusCondition(player: String, condition: String) = getPlayer
132
133     fun evalCronusCondition(player: Player, condition: String) = HookCronu
134
135     companion object {
136
137         val INSTANCE = Assist()
138
139     }
140
141
142 }
```

FUNCTIONS

菜单传参

启用

- 菜单传参默认是开启的，你也可以在菜单配置的选项中关闭

→ 选项 /menu/structure/options

传递

- 若菜单设置开启命令为 Example, 且开启参数功能
- 当玩家输入 /Example Agument1 Argument2 时，其菜单参数将设置为
  - [Argument1, Argument2] 2个
- 你也可以在开启命令、打开菜单动作中传递参数，或是通过设置参数动作二次更改参数
- 对于含有空格的参数，使用 `` 框起来

应用

- 菜单的参数可以当作 **变量** 来使用,
- {0} 对应第一个参数，{1} 对应第二个参数, 依此类推
- 对于 10个 以上的参数或是在部分特殊情况下，你也可以通过 PlaceholderAPI 变量来使用参数

→ PlaceholderAPI /hook/placeholderapi

物品特征

格式

- 同一个物品的不同特征匹配，用 , 分开
- 多个物品特征用 ; 分开

示例

```
material:DIAMOND,amount:64 ; material:EMERALD,amount:32
```

特征

- 材质: Material
- 损伤值: Data
- 数量: Amount
- 名称: Name
- 描述: Lore
- 头颅材质: Texture
- ModelData

脚本变量

格式

```
${js: <function>}
```

示例

```
# Lore 中
- '&8| &7ONLINE: &e%server_online% &6${js: utils.getOnlinePlayers().size()
```

注册命令

配置

settings.yml

```yaml
#
# 注册命令执行动作
# (增改命令需要重启服务器, TAB补全才能生效)
#
RegisterCommands:
  # 主命令
  openMenus:
    # 别称
    aliases: []
    # 权限
    permission: null
    # 无参数执行时
    execute:
      - 'tell: &7Argument Required!'
    # 参数及对应的反应
    arguments:
      example: 'open: example'
```

- 任何修改保存配置文件后都将自动生效
- 若新增命令，TAB 补全功能必须重启服务器才会有效

快捷绑定

配置

settings.yml

```yaml
#
# 快捷操作执行动作
#
Shortcuts:
  Offhand: 'open: Example'
  Sneaking-Offhand:
    - condition: 'hasPerm.trmenu.shortcut'
      execute: 'open: Example'
  Right-Click-Player: []
  Sneaking-Right-Click-Player: []
  PlayerInventory-Border-Left: []
  PlayerInventory-Border-Right: []
  PlayerInventory-Border-Middle: []
```

注意

- 触发动作后均执行 **反应**，且将取消事件
- 若不需要该项快捷动作，直接删除节点或者设置为空

→ 反应 /actions/reactions

模板创建

HOOK

PlaceholderAPI

Menus

```
%trmenu_menus%
```

> 返回服务器加载的菜单总数

Args

```
%trmenu_args_0%

%trmenu_args_0-5%
```

> 返回指定的菜单参数或范围参数拼接值

Meta

```
%trmenu_meta_Key%
```

> 返回对于 Key 值的 Meta 值

Data

```
%trmenu_data_Key%
```

> 返回对于 Key 值的 Data 值

Menu

```
%trmenu_menu_pages% # 当前菜单的总页数
%trmenu_menu_page% # 当前页码
%trmenu_menu_next% # 下一页的页码
%trmenu_menu_title% # 当前菜单的标题
```

> 返回玩家当前查看的菜单信息

EmptySlots

```
%trmenu_emptyslot_0% # 第一个空槽位
%trmenu_emptyslot_1% # 第二个空槽位

%trmenu_emptyslot_0_5-15% # 在 5-15 槽位范围内，第一个空槽位
```

> 返回玩家当前查看的菜单的空槽位信息

Js

```
%trmenu_js_<Expression>% # 执行脚本
```

> 返回 Js 脚本编译值

Cronus

Vault

PlayerPoints

HeadDatabase

Oraxen

SkinRestorer

API

TrMenuAPI

Events

MenuOpenEvent

```kotlin
package me.arasple.mc.trmenu.api.events

import io.izzel.taboolib.module.event.EventCancellable
import me.arasple.mc.trmenu.display.Menu
import org.bukkit.entity.Player

/**
 * @author Arasple
 * @date 2020/3/17 16:29
 */
class MenuOpenEvent(val player: Player, val menu: Menu, val page: Int, val

    fun isOpenedByBindings() = reason == Reason.BINDING_COMMANDS || reason

    enum class Result {

        /**
         * Unknow page etc.
         */
        ERROR_PAGE,

        SUCCESSED,

        DENIED,

        UNKNOWN

    }

    enum class Reason {

        RELOAD,

        PLAYER_COMMAND,

        CONSOLE,

        SWITCH_PAGE,

        BINDING_COMMANDS,

        BINDING_ITEMS,

        BINDING_SHORTCUT,

```

```
46              UNKNOWN
47
48          }
49
50      }
```

MenuCloseEvent

```kotlin
package me.arasple.mc.trmenu.api.events

import io.izzel.taboolib.module.event.EventCancellable
import me.arasple.mc.trmenu.display.Menu
import org.bukkit.entity.Player

/**
 * @author Arasple
 * @date 2020/7/7 9:31
 */
class MenuCloseEvent(val player: Player, val menu: Menu, val page: Int, va

    enum class Reason {

        CONSOLE,

        OP,

        PLAYER,

        MENU_RELOAD,

        SWITCH_MENU,

        SWITCH_BUKKIT_INVENTORY,

        SWITCH_PAGE,

        ERROR;

        fun isSwitch() = this == SWITCH_MENU || this == SWITCH_PAGE || thi

    }

}
```

MenuClickEvent

```kotlin
package me.arasple.mc.trmenu.api.events

import io.izzel.taboolib.module.event.EventCancellable
import me.arasple.mc.trmenu.api.inventory.InvClickType
import me.arasple.mc.trmenu.display.Icon
import me.arasple.mc.trmenu.display.Menu
import org.bukkit.entity.Player

/**
 * @author Arasple
 * @date 2020/7/6 21:44
 */
class MenuClickEvent(val player: Player, val slot: Int, val menu: Menu, va
```

MenuFactory